# خوارزمية ديكسترا (Dijkstra Algorithm)

هي خوارزمية لإيجاد الطريق الأقصر (shortest path) من عقدة إلى عقدة أخرى أو من عقدة إلى جميع العقد الاخرى في البيان.

**ملاحظة:** تتعامل هذه الخوارزمية مع البيان الموزون.

**مبدأ عمل الخوارزمية:**

**1-** ننشئ مصفوفة تسمى sptSet والتي تحوي على العقد التي تم زيارتها وضمها لشجرة المسار الأقصر، بالطبع تم حساب المسار الأقصر لهذه العقد. ابتداءاً تكون جميع العقد غير معالجة أي أن قيم عناصر المصفوفة هو false.

**2-** ننشئ مصفوفة مسافات dist تحوي على الأوزان (المسافات) من العقدة المصدر إلى باقي العقد وتكون بالبداية قيم هذه المسافات هي لانهائية (INFINITE)، ثم توضع المسافة بين عقدة المصدر مع نفسها هي صفر. وذلك لأنه تم معالجتها أولاً.

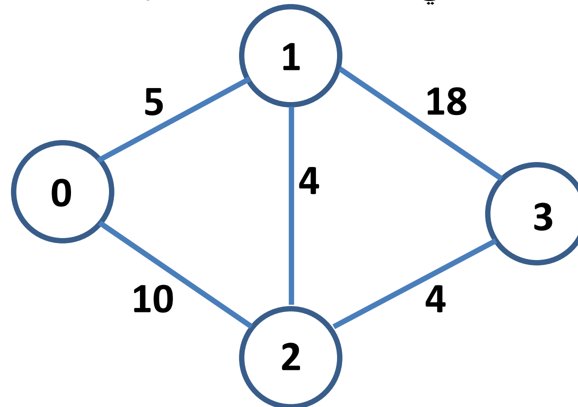
**3-** طالما أن المصفوفة sptSet لا تحوي جميع العقد المعالجة: قم بالتالي:

**A.** التقط عقدة u والتي هي غير موجودة في المصفوفة sptSet ولها أقل قيمة مسافة، يستدعى التابع minDistance لحساب أي عقدة لها أقل مسافة.

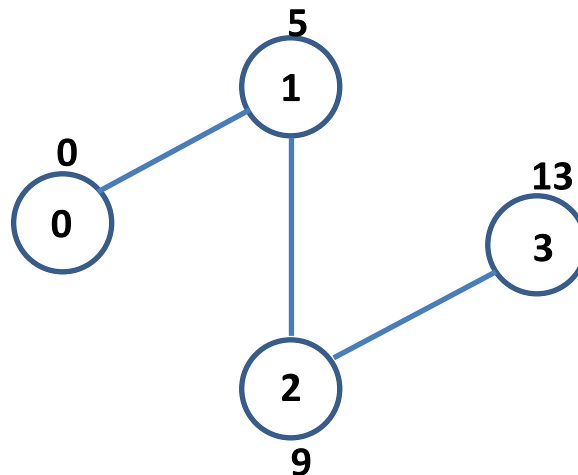
**B.** قم بضم العقدة u إلى المصفوفة sptSet أي اجعل قيمتها True.

**C.** قم بتحديث وحساب قيم المسافات لجميع جوار العقدة u والذين سنسميهم v. إذا كان مجموع المسافة من عقدة المصدر للعقدة u مع وزن الرابط بين u و v أقل من وزن v الحالي عندها قم بتحديث الطريق للعقدة v.

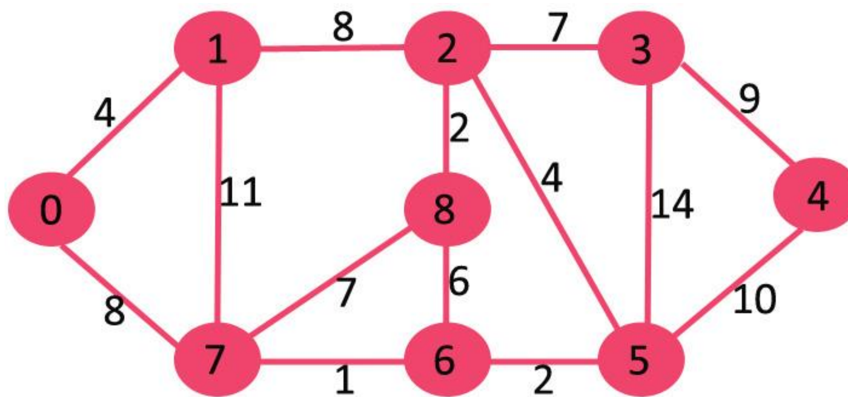
**مثال1:** طبق خوارزمية الطريق الأمثل للبيان التالي لحساب المسافة الأقصر بين العقدة 0 وجميع العقد الأخرى:



**الحل:**




**مثال2:** طبق خوارزمية الطريق الأمثل للبيان التالي لحساب المسافة الأقصر بين العقدة 0 وجميع العقد الأخرى:




**الحل:**

| | |
|---|---|
| | A. رسمة التنفيذ بعد إلتقاط u=0<br>B. وضمها إلى المصفوفة sptSet بجعلها True<br>C. وحساب وتحديث قيم المسافات لجميع جوار العقدة 0 وهما 1 و 7 |
| | A. رسمة التنفيذ بعد إلتقاط u=1<br>B. وضمها إلى المصفوفة sptSet بجعلها True<br>C. وحساب وتحديث قيم المسافات لجميع جوار العقدة 1 وهو 2 |

| | |
|---|---|
| | A. رسمة التنفيذ بعد إلتقاط u=7<br>B. وضمها إلى المصفوفة sptSet بجعلها True<br>C. وحساب وتحديث قيم المسافات لجميع جوار العقدة 7 وهما 6 و 8 |
| | A. رسمة التنفيذ بعد إلتقاط u=6<br>B. وضمها إلى المصفوفة sptSet بجعلها True<br>C. وحساب وتحديث قيم المسافات لجميع جوار العقدة 6 وهما 5 و 8 |
| | و هكذا حتى تصبح الرسمة على الشكل التالي: |

**مثال:** اكتب كود خوارزمية ديكسترا للبيان السابق في المثال 2 بلغة ++C، ثم احسب المسار الأقصر بدءاً من العقدة 0 إلى جميع العقد في البيان.

```
// A C++ program for Dijkstra's single source shortest path algorithm.
// The program is for adjacency matrix representation of the graph
#include <iostream>
using namespace std;

// Number of vertices in the graph
#define V 9

// A utility function to find the vertex with minimum distance value, from
// the set of vertices not yet included in shortest path tree
int minDistance(int dist[], bool sptSet[])
{

      // Initialize min value
      int min = INT_MAX, min_index;

      for (int v = 0; v < V; v++)
            if (sptSet[v] == false && dist[v] <= min)
                  min = dist[v], min_index = v;
```

```
	return min_index;
}

// A utility function to print the constructed distance array
void printSolution(int dist[])
{
	cout <<"Vertex \t Distance from Source" << endl;
	for (int i = 0; i < V; i++)
		cout << i << " \t\t"<<dist[i]<< endl;
}

// Function that implements Dijkstra's single source shortest path algorithm
// for a graph represented using adjacency matrix representation
void dijkstra(int graph[V][V], int src)
{
	int dist[V]; // The output array. dist[i] will hold the shortest
	// distance from src to i

	bool sptSet[V]; // sptSet[i] will be true if vertex i is included in
shortest
	// path tree or shortest distance from src to i is finalized

	// Initialize all distances as INFINITE and stpSet[] as false
	for (int i = 0; i < V; i++)
		dist[i] = INT_MAX, sptSet[i] = false;

	// Distance of source vertex from itself is always 0
	dist[src] = 0;

	// Find shortest path for all vertices
	for (int count = 0; count < V - 1; count++) {
		// Pick the minimum distance vertex from the set of vertices not
		// yet processed. u is always equal to src in the first
iteration.
		int u = minDistance(dist, sptSet);

		// Mark the picked vertex as processed
		sptSet[u] = true;

		// Update dist value of the adjacent vertices of the picked
vertex.
		for (int v = 0; v < V; v++)

			// Update dist[v] only if is not in sptSet, there is an edge
from
			// u to v, and total weight of path from src to v through u
is
			// smaller than current value of dist[v]
			if (!sptSet[v] && graph[u][v] !=0
```

```
                    && dist[u] + graph[u][v] < dist[v])
                    dist[v] = dist[u] + graph[u][v];
    }

    // print the constructed distance array
    printSolution(dist);
}

// driver program to test above function
int main()
{

    /* Let us create the example graph discussed above */
    int graph[V][V] = { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },
                        { 4, 0, 8, 0, 0, 0, 0, 11, 0 },
                        { 0, 8, 0, 7, 0, 4, 0, 0, 2 },
                        { 0, 0, 7, 0, 9, 14, 0, 0, 0 },
                        { 0, 0, 0, 9, 0, 10, 0, 0, 0 },
                        { 0, 0, 4, 14, 10, 0, 2, 0, 0 },
                        { 0, 0, 0, 0, 0, 2, 0, 1, 6 },
                        { 8, 11, 0, 0, 0, 0, 1, 0, 7 },
                        { 0, 0, 2, 0, 0, 0, 6, 7, 0 } };

    dijkstra(graph, 0);
    system("pause");
    return 0;
}
```

الخرج هو

```
Vertex   Distance from Source
0                0
1                4
2                12
3                19
4                21
5                11
6                9
7                8
8                14
```

**تمرين:** ما الذي سيتم تعديله في كود خوارزمية ديكسترا السابق بحال كانت أسماء العقد هي محارف وليست أرقام؟